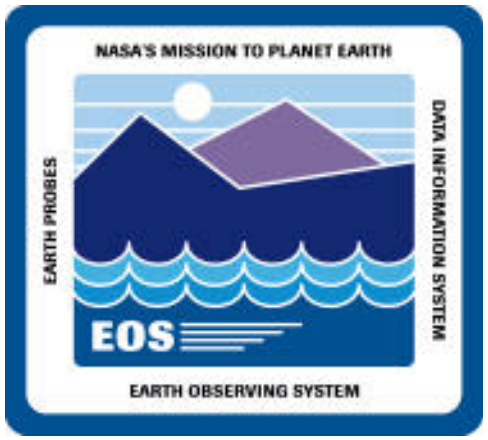
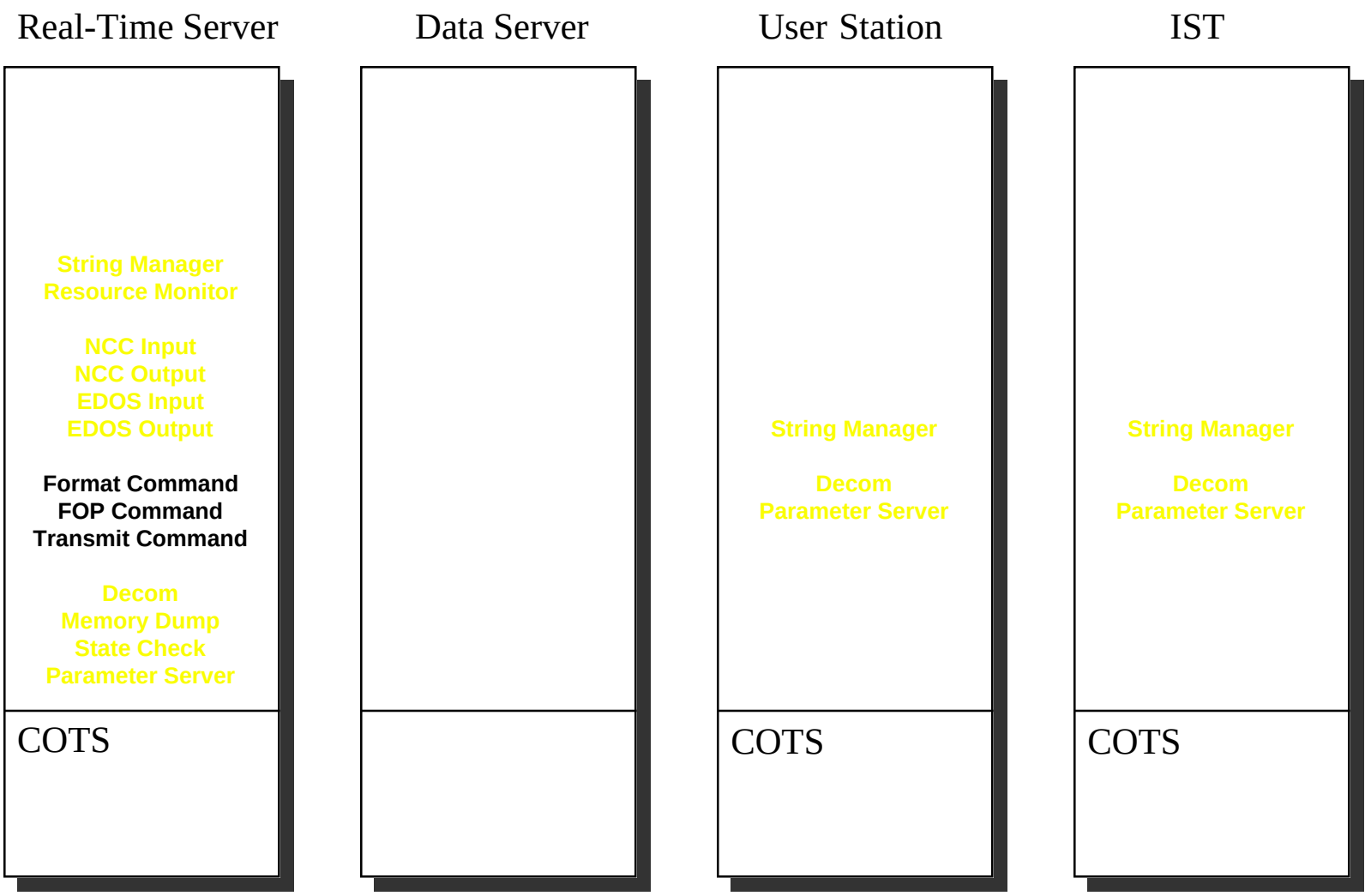


Command Subsystem

J.B. Wilson

17 October 1995

Process-to-Hardware Mapping

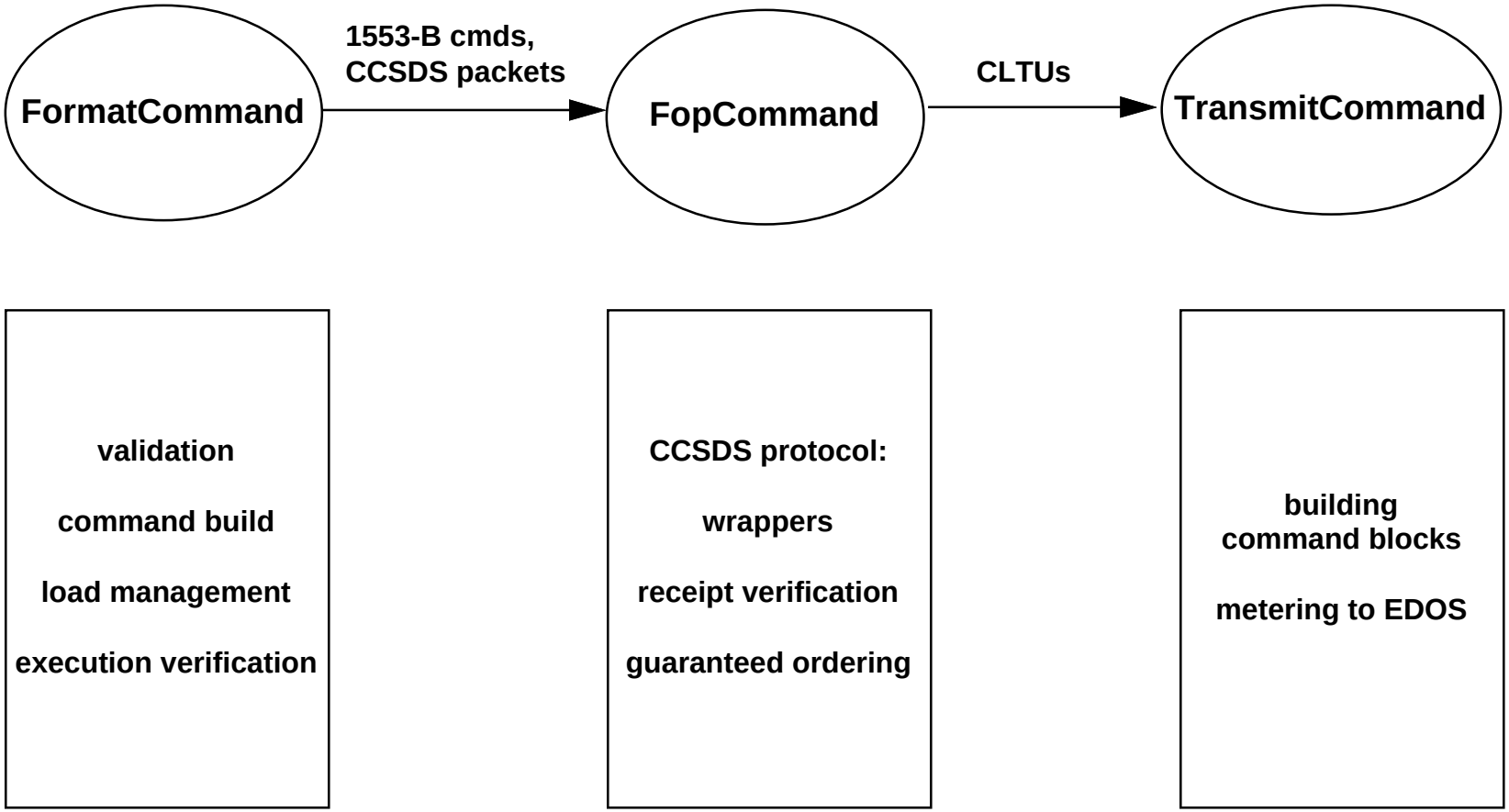


Subsystem Objectives

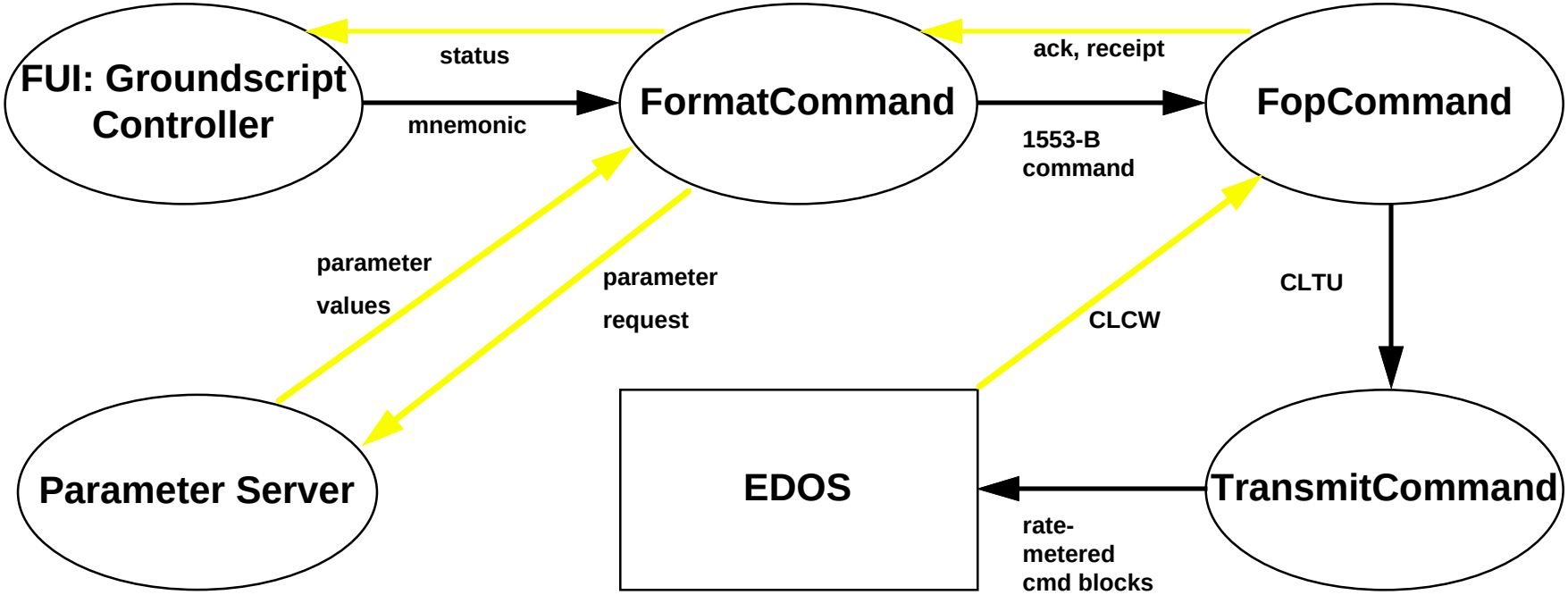
- **Goal was to design in the flexibility necessary for multi-mission re-use**

- **Keys to flexibility are:**
  - **Modularity**
    - **Encapsulation by process**
      - **S/C specific**
      - **Uplink protocol (COP-1)**
      - **Metering of uplink data**
    - **Natural extension of object-oriented approach**
  - **Database driven**
    - **Building of commands, command type, etc.**

Overview of Command Subsystem



Real-Time Command Processing Flow



Real-Time Command Processing

An overview of how a real-time command gets through the Command Subsystem:

VALIDATION

- **User authorization check**
- **Mnemonic validation (access CMD definition from database)**
- **Check submnemonics & specified values**
- **Prerequisite checking**
- **Critical command prompting**

BUILD - 1553-B format

- **Command destination**
- **Command descriptor**
- **Command data - fixed and variable (submnemonic specified)**

Real-Time Command Processing (cont.)

CCSDS PROTOCOL

- **Uplink processing:**
  - **CCSDS packet**
  - **Transfer frame**
  - **CLTU**
- **Receipt verification:**
  - **CCSDS Command Operations Procedure:**

    CLCWs provide uplink status

    COP-1 provides “go back n” automatic retransmission

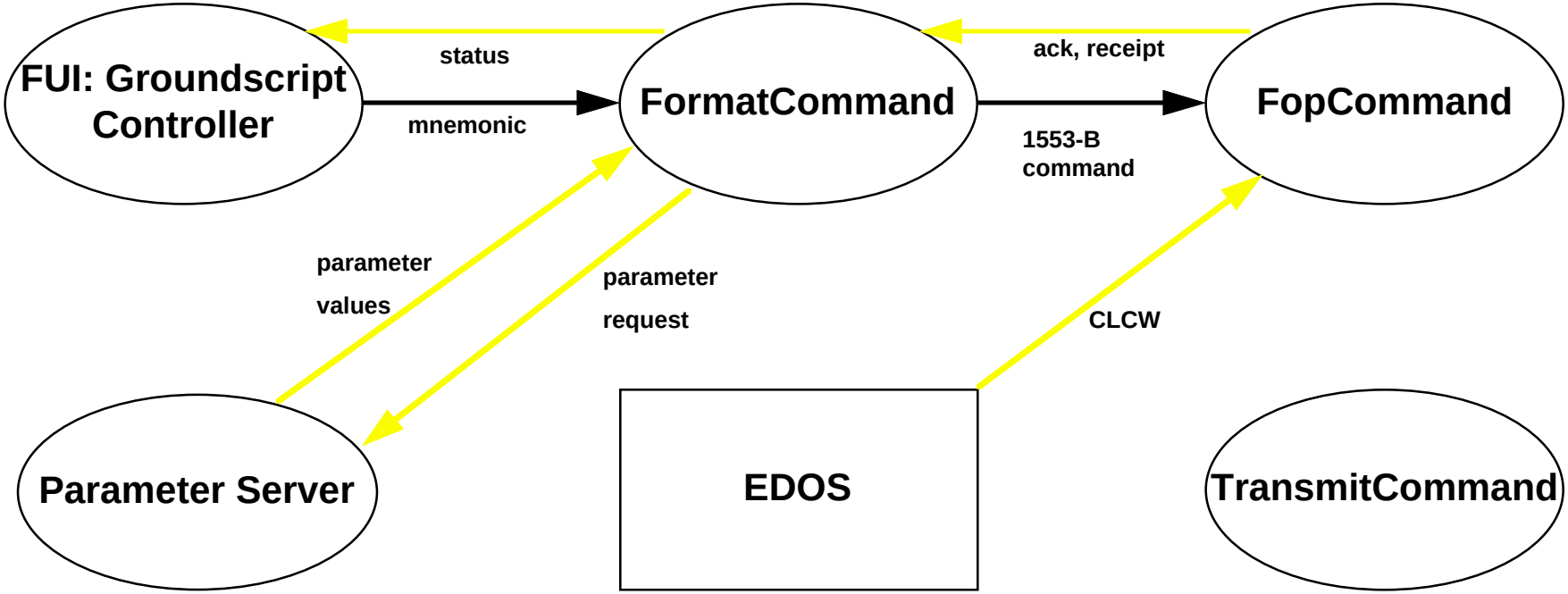
COMMAND TRANSMISSION

- **Command transmission metering to adhere to requested data rate**

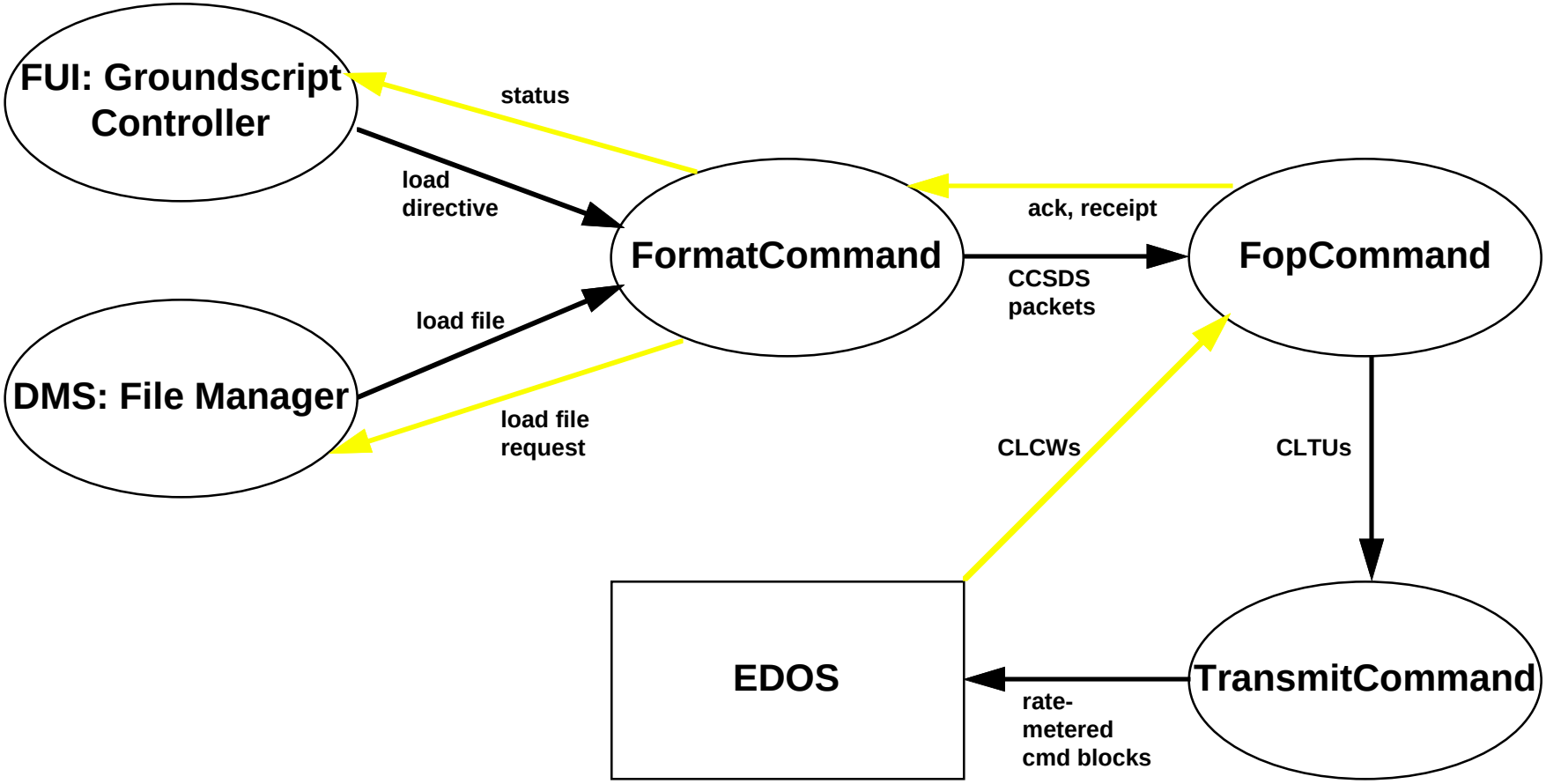
TELEMETRY VERIFICATION

- **Verification of the execution of commands**

Real-Time Command Processing Flow (Backup String)



Load Processing Flow



Load Processing

An overview of how a load gets through the Command Subsystem:

VALIDATION

- **User authorization check**
- **Load parameter validation - time window & spacecraft Id**
- **Partitioned loads: correct sequence ensured**
  - **Can be overridden**
- **Prerequisite checking**
- **Critical command prompting for loads containing critical commands**

CCSDS PROTOCOL

- **Uplink processing:**
  - **CCSDS packet (this is the format of the data in the load file)**
  - **Transfer frame**
  - **CLTU**

Load Processing (cont.)

CCSDS PROTOCOL (cont.)

- **Uplink verification:**
  - **CCSDS Command Operations Procedure:**
    - **CLCWs provide uplink status**
    - **COP-1 provides “go back n” automatic retransmission**

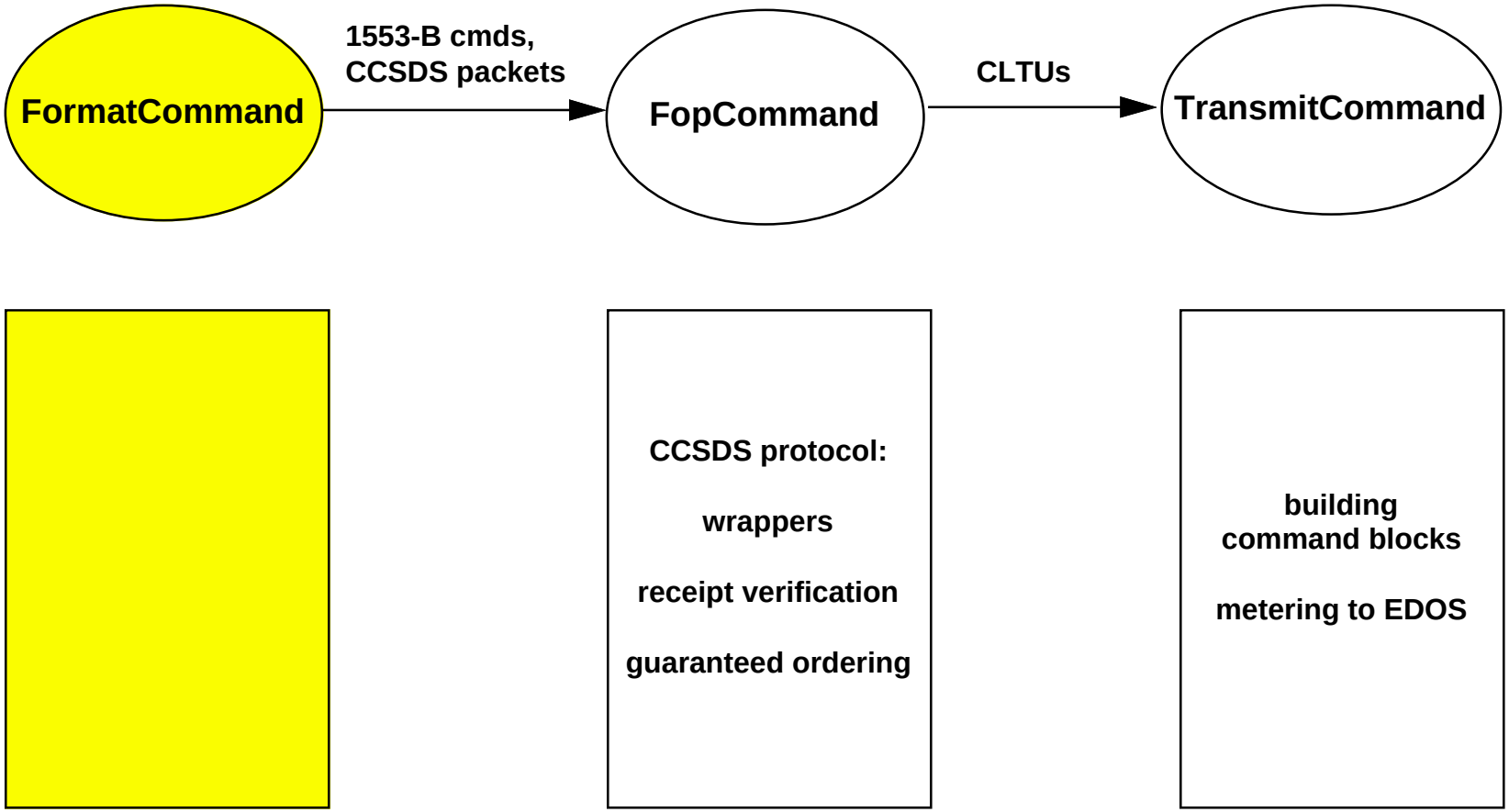
COMMAND TRANSMISSION

- **Command transmission metering to adhere to requested data rate**

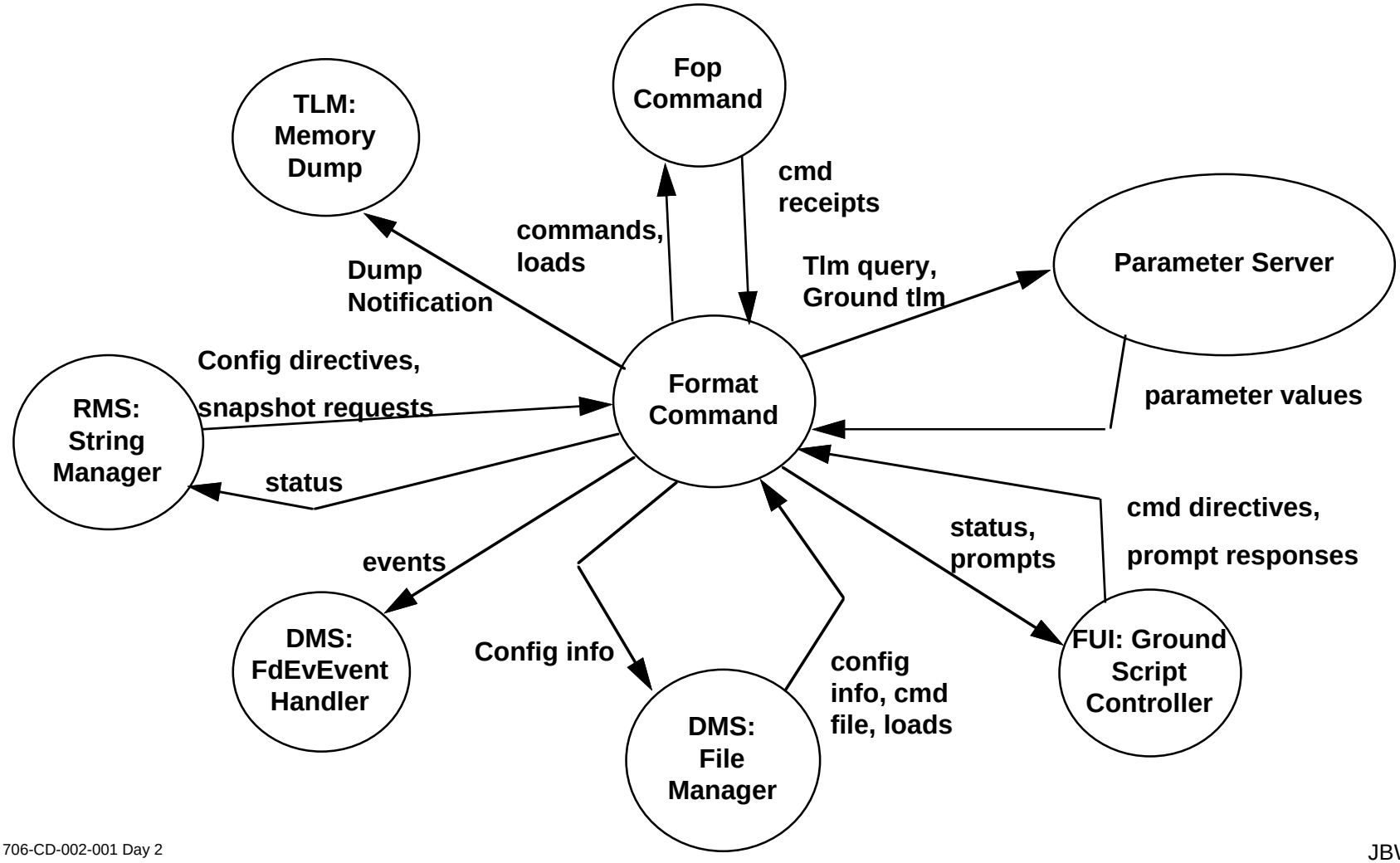
TELEMETRY VERIFICATION

- **CRC check confirmation of successful installation of load onboard**

FormatCommand Process



FormatCommand Context Diagram



Key Features of FormatCommand

Flexibility: result of database-driven design. The Command Operational Data File stores:

- **Parameters for command validation**
- **Telemetry parameters for command execution verification**
- **Coefficients for reverse EU conversion**

Improve speed: the data file is loaded into memory to reduce access time

Re-use: Real-time commands, stored commands and loads are derived classes of a base command class:

- **Future enhancement or modifications can be done easily**
- **These classes can be re-used in future projects.**

FormatCommand: Parameters

Telemetry parameters defined in Command Operational Data File:

- **Command mnemonics**
- **Command submnemonics**
- **Submnemonics default values**
- **Submnemonics ranges**
- **Polynomial coefficients for reverse EU conversion**
- **Parameters for prerequisite check (PIDs, Ranges, prereq flag)**
- **Parameters for execution verification (PIDs, Ranges, Time out)**
- **Command destination and command descriptor**
- **Critical flag**
- **Dump command flag**

FormatCommand: Parameters (cont.)

Command parameters defined in CMS Load Catalog file:

- **Parameters for load validation**
- **Parameters for load execution verification**

Command parameters controlled by users:

- **Prerequisite flag (enable/disable)**
- **Submnemonic values**

FormatCommand: Validation

Validation of Real-time Commands: performing the following checks:

- **User authorization check**
- **Command mnemonic exists in database**
  - **For dump command, notify Telemetry Subsystem; send out an event message if the dump command is not telemetry verified**
- **All user-entered submnemonics are validated via the database**
- **All submnemonics have either default values or user-entered values**
  - **Perform reverse EU conversion**
- **User-entered values (int or real) are within ranges defined in database**
- **Prerequisite check (raw, decoded, converted):**
  - **Check that prerequisite parameter values are non-static**
  - **Check that prerequisite parameter values are within ranges defined in database**
  - **Provides ability to disable prerequisite check**
  - **Provides ability to override prerequisite check failure**

FormatCommand: Validation (cont.)

- **Criticality check:**
  - **Check if command is critical and, if so, prompt for permission.**

Validation of Hex commands:

- **User authorization check**
- **All hex commands treated as critical commands, user is prompted**

Validation of Stored commands:

- **User authorization check**
- **Check that command mnemonic exists in database**

Validation of Loads:

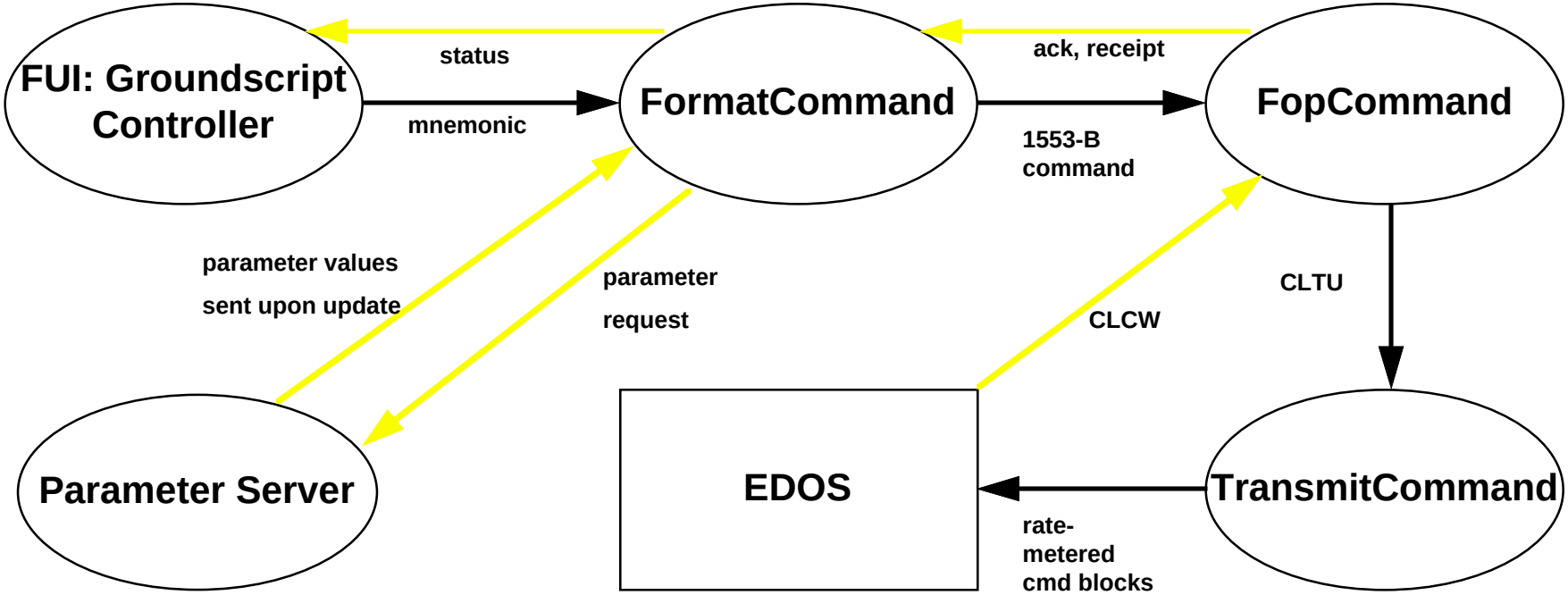
- **User authorization check**
- **Check spacecraft ID**
- **Check current time against time window allowed for a load/partition**
- **Check uplink time of preceding partitions**
- **Check prerequisite as in real-time command**
- **Prompt for critical permission if a load contains a critical command**

FormatCommand: Build

Build:

- **Real-time commands: format into 1553-B before forwarding to FopCommand. A 1553-B structure contains command destination, command descriptor and optional command data**
- **Hex commands: user enters commands in 1553-B format**
- **Loads: CMS provides loads in CCSDS packet format**

FormatCommand: Execution verification



FormatCommand: Execution Verification

Execution verification design features:

- **Fast: Notification-upon-update service of Parameter Server allows verification as soon as possible**
- **Efficient: All commands having common verification parameter are grouped together**
- **Extendable: there is no limit in the number of commands associated with a verification parameter**

Current design for execution verification:

- **FormatCommand maintains a list of verification parameters**
- **Verification parameter values are supplied via Parameter Server**
- **Associated with each element in the parameter list is a list of commands**

FormatCommand: Execution Verification (cont.)

Execution verification of a real-time or a stored command:

- **During the time window, check telemetry value (raw, decoded or converted) for the command against ranges defined in database**

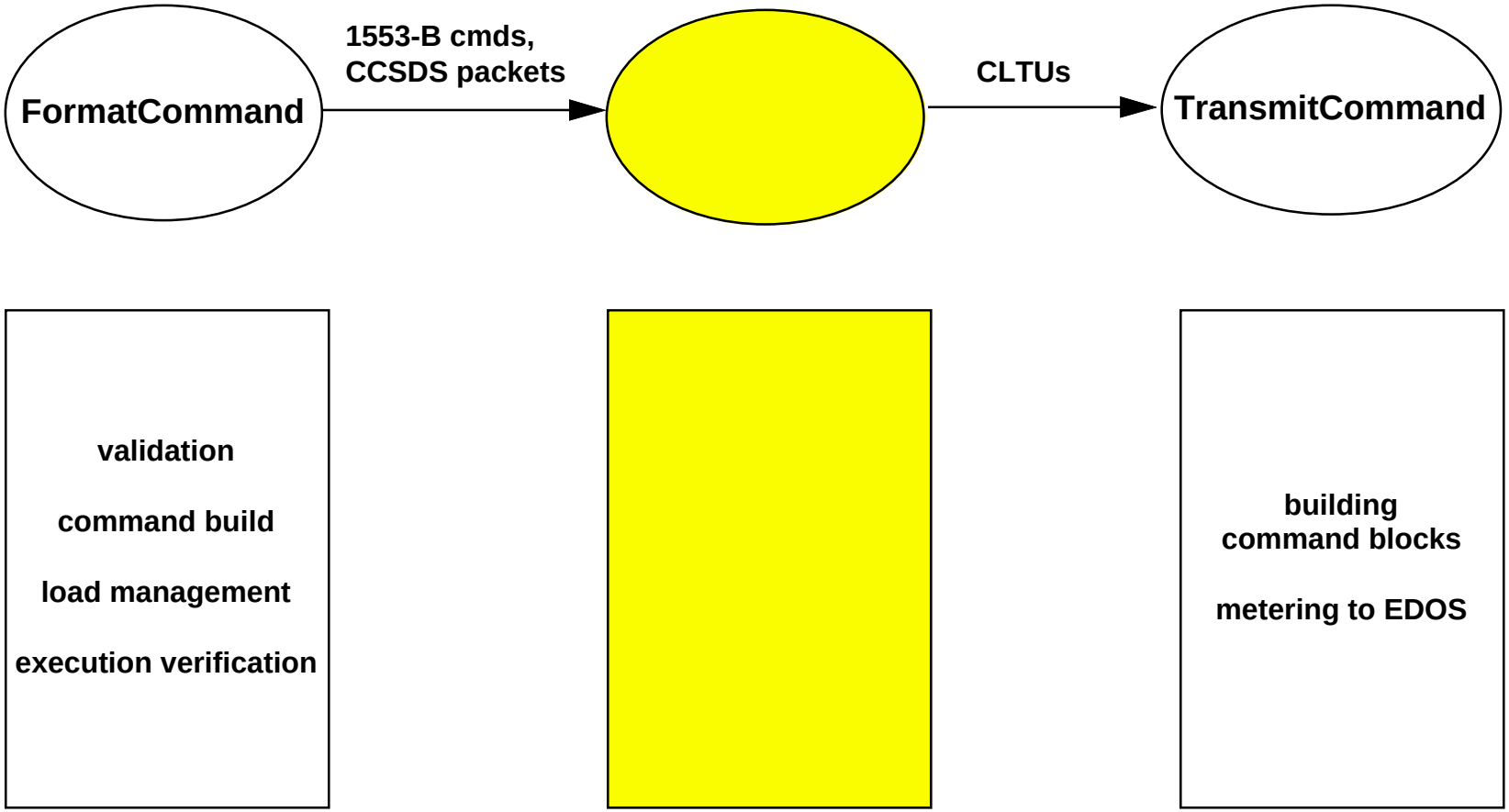
Execution verification of a Hex command:

- **No verification parameters defined in the Data file**

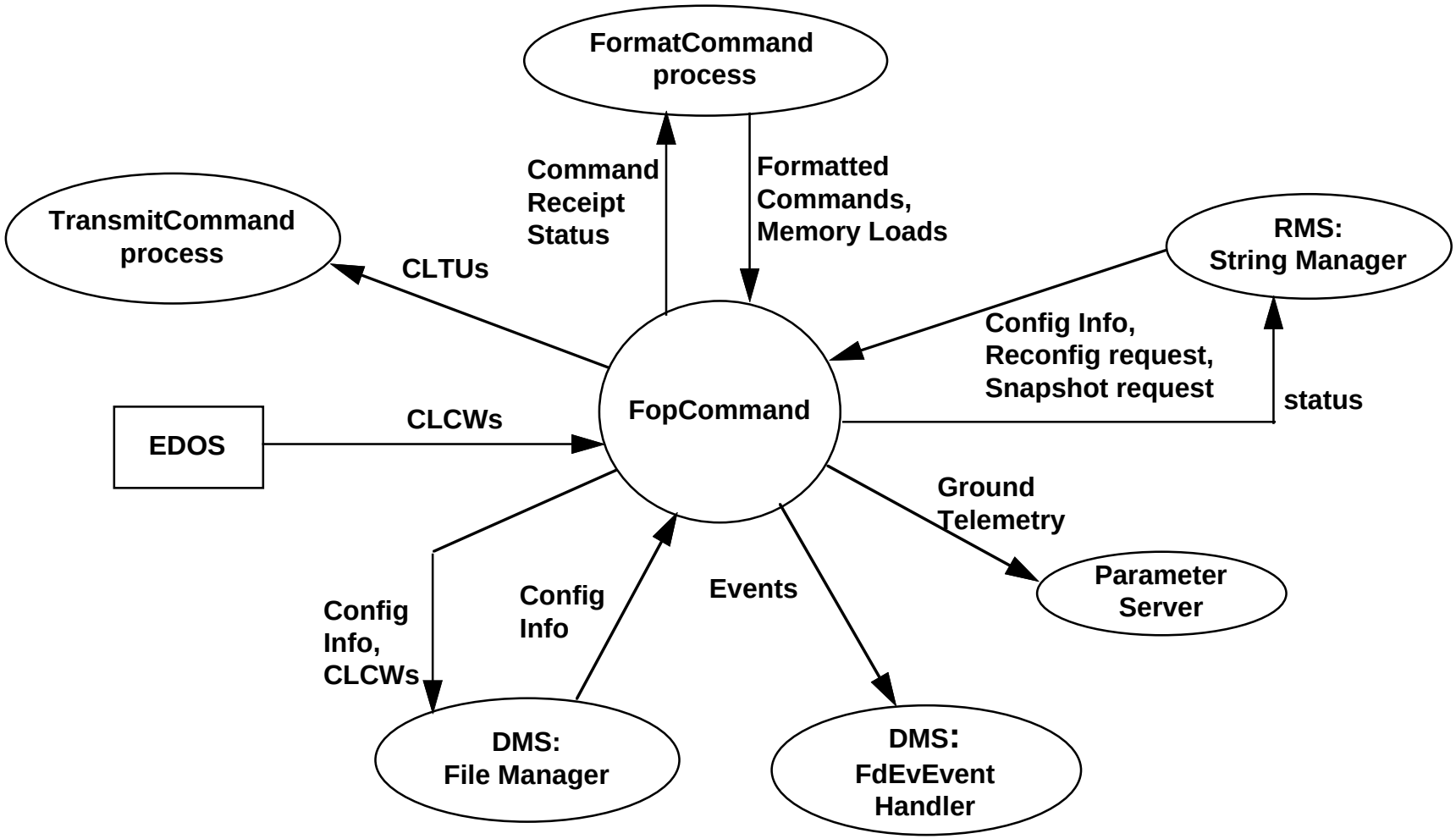
Execution verification of a Load/Partition:

- **Check down linked CRC or flag for the load/partition**
- **CRC values is broadcasted as ground telemetry for display**
- **Notify CMS of load/partition uplink time**

FopCommand Process



FopCommand Context Diagram



Key Features of FopCommand

Modular design : all aspects of CCSDS protocol are addressed in FopCommand

- **Uplink protocol can be changed without affecting other processes**

FopCommand design :

- **Based on state machine method: the design mirrors the CCSDS state table**
- **Enhance maintainability**
- **Easy to accommodate future enhancement:**
  - **When segmentation layer is added, multiple instance of state machine can be created, one for each virtual channel.**
- **New functions can easily be added to provide Expedited Service (BD Service) and the segmentation layer**

FopCommand Analyses

- In the FopCommand prototype, table look up method was tried
  - It was determined to be difficult to partition behavior for different states
  - State class gets very complicated: hard to maintain
- State-machine method was implemented:
  - It localizes state-specific behavior for each state
  - New transitions can be added to a particular state without affecting others.
- A CRC (Cyclic Redundancy Check) class was developed to handle CRC calculation:
  - The Class calculates CRC-BCH and CRC-CCITT code correctly (verified with Lockheed Martin)
  - Sub class can be easily added to incorporate other CRC calculation schemes.

COP-1 Displayable Items

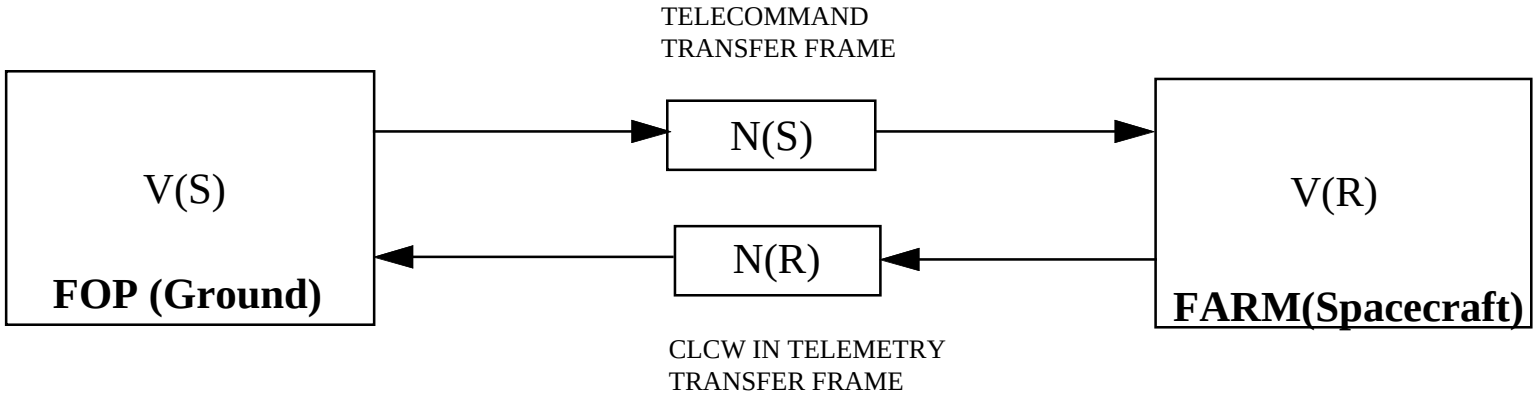
- **FOP**
  - **Ground transmitter sequence number**
  - **Sliding window width**
  - **Transmission limit (default: 3 times)**
  - **T1_Initial (default: 7 sec)**
  - **Time-out type (default: Alert)**
- **CLCW**
  - **Lockout flag**
  - **Retransmit flag**
  - **Wait flag**
  - **Next Expected Sequence Number**

COP-1 User Configurables

Directives from the user control Fop protocol behavior and can:

- **Terminate AD service via “Terminate AD” directive**
- **Set ground transmitter sequence number (used to synchronize COP-1)**
- **Set Fop Sliding Window Width**
- **Set Fop Transmission Limit**
  - **Setting Transmission Limit to 0 effectively turns off retransmission**
- **Set Fop T1_Initial (used to initialize timer)**
- **Set Fop Time-out Type (Alert or Suspend), which specifies what action to take when time-out occurs**
- **Resume AD service after it is suspended**

FopCommand: AD Service



FOP: Frame Operation Procedure
Provide Sequence-Controlled Service (AD Service) at the ground site.

FARM: Frame Acceptance and Reporting Mechanism
Provide Sequence-Controlled Service (AD Service) at the spacecraft site.

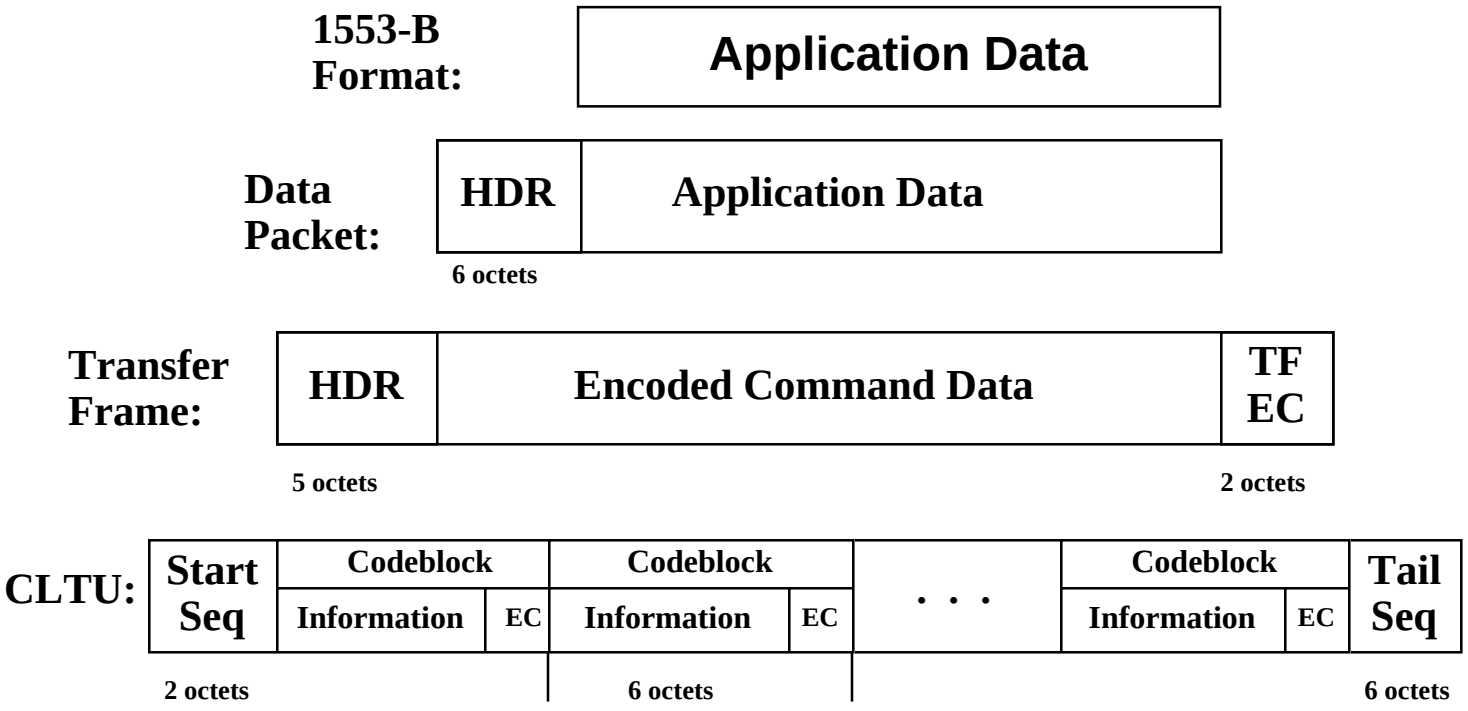
V(S): Transmitter Frame Sequence number.

N(S): Frame Sequence Number in the TC Frame Header.

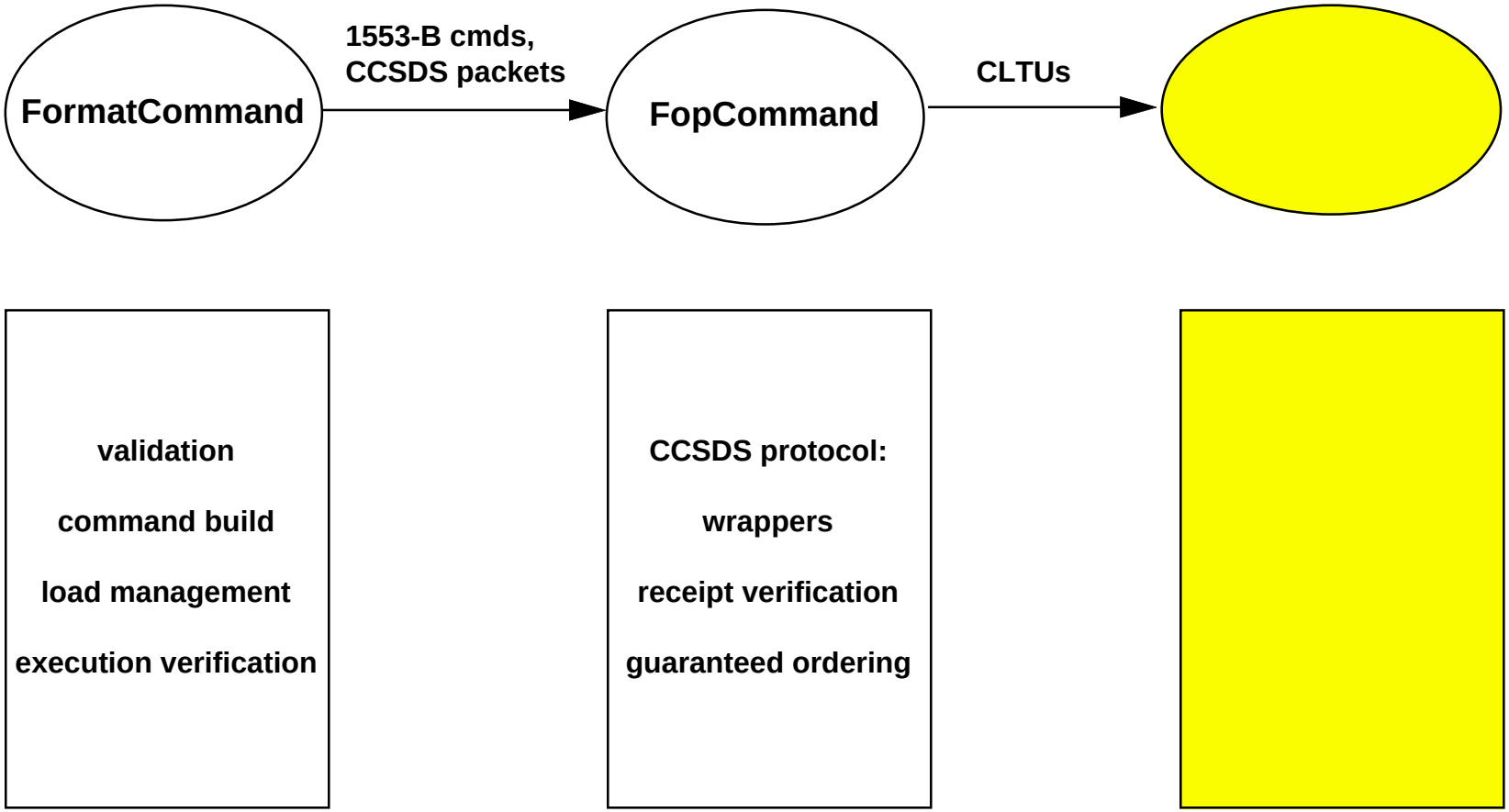
V(R): Receiver Frame Sequence Number.

N(R): Next Expected Frame Sequence Number contained in the CLCW.

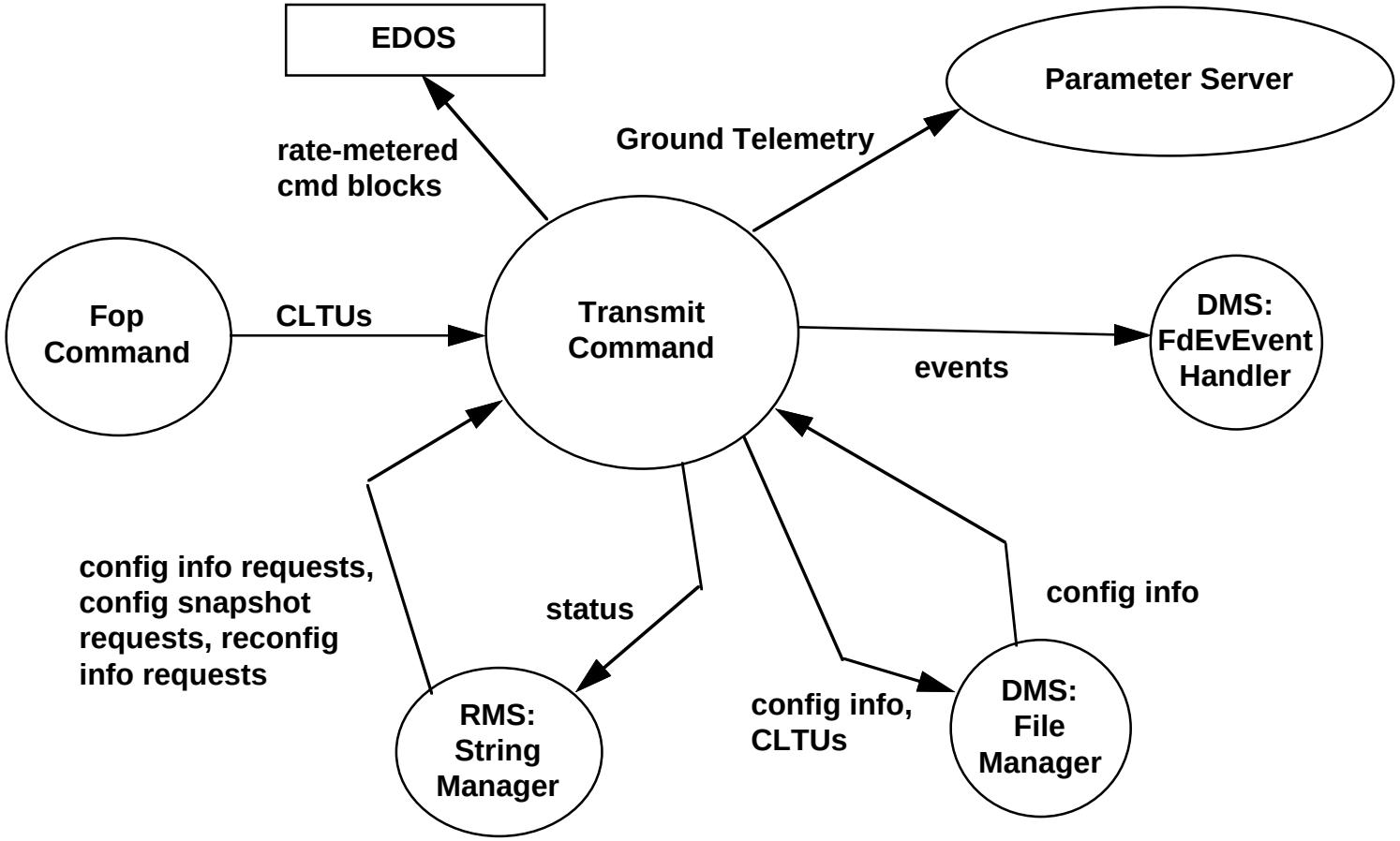
FopCommand: Data structures



TransmitCommand Process



TransmitCommand Context Diagram



TransmitCommand: Metering

Design features:

- **Efficient : the next command block is sent out as soon as possible so that full bandwidth is utilized**
- **Modular : different transmission mechanism can be accommodated in TransmitCommand without affecting other processes**
- **Re-use: the TransmitCommand process can be “plugged into” future missions**

Metering method:

- **Package all CLTUs received from FopCommand into command blocks and then put them into internal queue to wait to be sent**
- **Meter command blocks from internal queue to EDOS according to uplink rate (125 bps, 1 kbps, 2 kbps, or 10 kbps)**
- **Derive uplink rate from uplink path and display to users via ground telemetry**

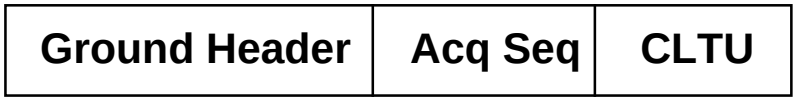
AM-1 PLOP

PLOP-1

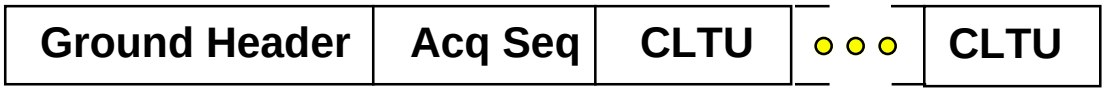
Real-time command and loads

PLOP-2

Real-Time command



Loads



TransmitCommand: Transmission Protocol

CCSDS PLOP function moved to the EOC from EDOS simplifies the end-to-end system

- **Reduces interface testing**
- **Provides EOC more control of uplink**

PLOP-2 allows better bandwidth utilization

- **Acquisition sequence is applied to blocks of CLTUs instead of to each one**
- **Block size is user-adjustable through configuration file**

TransmitCommand: Transmission Protocol (cont.)

Two transmission protocols : PLOP-1 and PLOP-2.

PLOP-1:

- **No difference between handling of real-time commands and loads**
- **Each UDP packet contains a ground header, an acquisition sequence, a single CLTU, and a gap**
- **AM-1 requires gap of 8 octets of all zero**

PLOP-2:

- **No gap is required**
- **Real-time Commands:**
  - **Each UDP packet contains a ground header, an acquisition sequence and a single CLTU**
- **Loads:**
  - **Each UDP packet contains a ground header, an acquisition sequence and 1 - 50 CLTUs**